01

Build 10 User Friendly Apps with Flutter

Bootcamp

Offline Course

Like Comment Share

Save

What we Provide

Flutter

Bootcamp
Offline Course

1. 8 Applications
2. 2 Major Projects
3. Internship
4. Daily widgets
5. Weekly Assignments
6. Quiz
7. Doubt solving sessions
8. Practical sessions
9. Completion certificate

Link in bio

www.core2web.in

02

1 Quiz Application

(e.g. Moodle, Duolingo)

2 To-do List Application

(e.g. Notes, Trello)

3 Expense Manager

(e.g. Monito, Khatabook)

4 E-Commerce App

(e.g. Myntra, Amazon)

5 Chat Application

(e.g. WhatsApp, Telegram)

6 Coffee Shop App

(e.g. CCD, Starbuks)

7 Food Delivery App

(e.g. Zomato, Swiggy)

8 Social Media

(e.g. Facebook, Instagram)

9 Flutter Flame

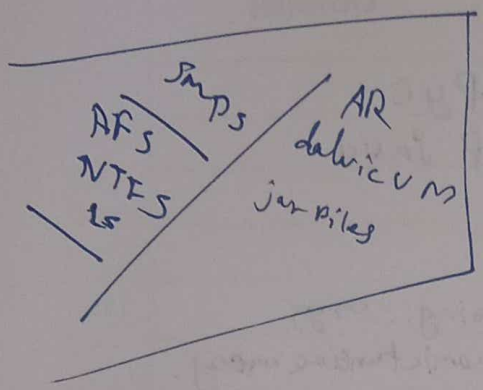
(e.g. 2D Game Apps)

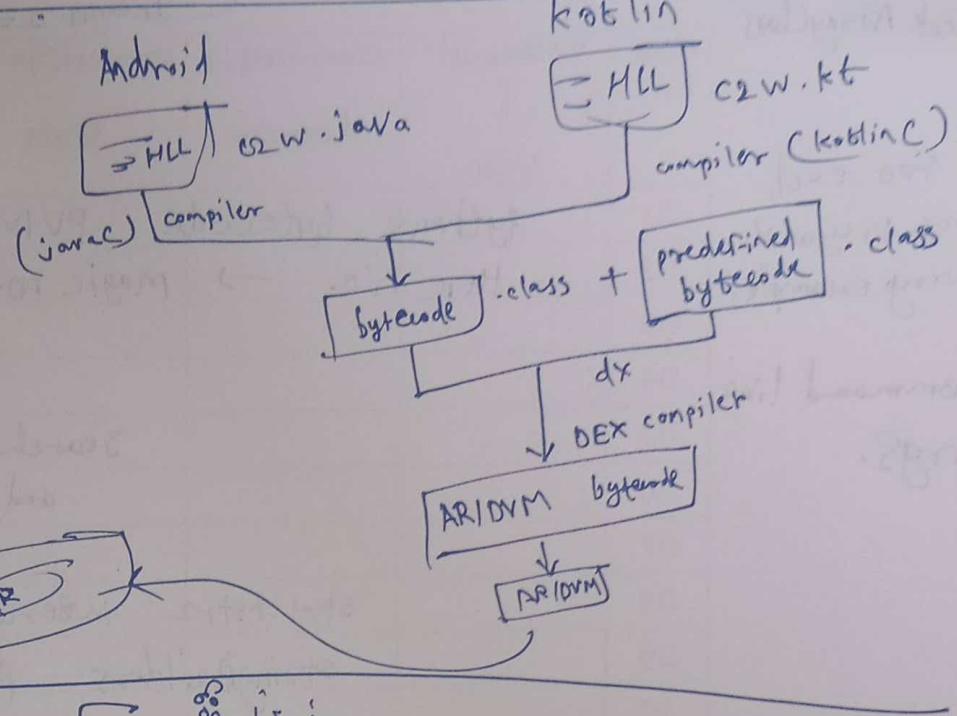
10 Flutter Web

(e.g. Netflix, Amazon)

॥ श्री ॥

ctrl + ~ to multi windows in VS code





Q. try -> reverse no. convert to string with string buff/build reverse();

"Compiler checks concept too"

is java gives garbage value?

Q. can for loop have 2 condⁿ? } check

task -> All type of number series try - Fibonacchi, Factorial, Prime, Perfect, Armstrong { count, series print, validaⁿ. palindrome check. reverse no.

try -> simplified output recompile

Q. with -Xdiags:verbose

try: Armstrong length find with tostring conversion & then use length() funⁿ.

try -> Array value input/output with BufferedReader InputStreamReader.

☆ { exceptions are run-time }

int arr[];
int arr[];
int arr2[] = new int[] {10,20...};
int arr1[] = {10,20...};
↳ internally goes same.

try -> system.
↓
Sout(identityhashcode(arr[0]));
Sout(arr);

check -> { integrals: int + character }

we may only gives size or list at a time

int arr2[] = new int[5] {10,...}; • error

Xcode ide

mac

garry kilder

Install

F/W tool — Flutter engine — Dart — predefined lib. → App/n built

1973
84
92
94
97
2000
01
07
08
11

unix os → 1969
→ C - 1972
→ rewrite - 1973

3 April 1973 → 1st phone → Martin Cooper motorola

Prototype
Actual → 1984
rivalry → Bell Lab.

Name → dynatac 8000x

1982 - car Phone
nokia

Mouse
↳ mother of all demo

1992 - 1st smartphone ← check...
IBM SPCC Simon Personal computer)
• touch screen with stylus
• 1994

1997 - First game - snake
- Nokio
- 6110

2000 -
↳ First camera phone
Japan → J-SH04
→ sharp comp. name
→ 0.1 mp

2001 -
↳ 1st internet & mobile connecn
→ 3G

2004 - Facebook / youtube
2005 - youtube
2006 - Acquire google
2005 → same about android

2007 - iphone 1.0
↳ Steve Jobs → 1955 birth
→ 1976 Apple inc.
↳ 3 names: steve Jobs, steve wozniak, Ronald wayne
1984 → Macintosh (Mac OS)

1985 → comp. exit
1985-86 → Next Comp
1986 → Acquire (Pixar Animath)
1997 → Next by Apple aquire

check → ios / Android version

2008
- 1st Android OS
- HTC T1-mobile G1 (HTC dream)
- Android (Google) → Java & xml
  2003 → Andy rubin & Team
- Actual Focus on digicam
- Android os 1.0
  C to P

Yahoo, Google
- HTC
- LG
- Motorola
- Samsung

OHA - Tie up
open Handset alliance

Nokia with windows

Android version:
cupcake, donut, eclair, froyo, GingerBread, HoneyComb, IceCreameSandwitch, JellyBin,
kitkat, Lollypop, Marshmello, Nogut, Oreo, Pie
Android 10, 11, 12, 12L, 13, 14, 15

Switch

APK → Android package kit
gradle build
linux dexfros.

Android OS, iOS, windows Phone OS, Symbian, Palm OS,
(windows Phone)
Blackberry OS → Blackberry APP world
UWP
KaiOS, Tizen OS, HarmonyOS
(Huawei)

Amb

colorOS → oppo
oxygenOS → oneplus

metro, modern UI
Skype

iOS → icloud
→ facetime

Mobile H/W: monitor screen (Display), Processor, Battery

- Camera, sockets
  - charging
  - headphone (optional)
- Audio components,
  - microphone
  - speaker
- sensors
- storage
  - internal
  - SD card
- Accelerometer
- Battery
  - Lithium polymer battery
  - Lithium-ion
- SIM slot (Subscriber identity module)
- Light
- Network components
  - Bluetooth
  - wifi
- Button
- RAM
- chasis
- components
  - IC, bus

→ This all are API

system call: kernel call in OS

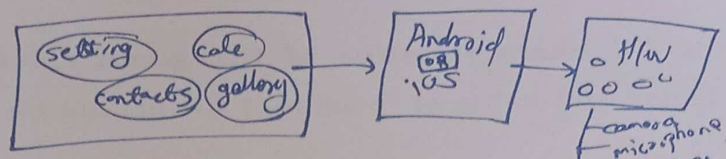
OS → kernel + utilities

How mobile apps works ?

Types of apps

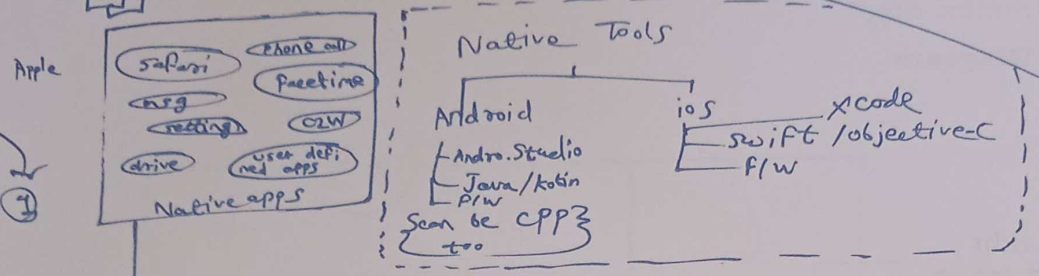
1. Try install diff. OS
2. Git repo maintain
3. Tools
   - IDE – Xcode

• Native apps: strictly bind to OS.



{check diff. native apps. for OS's}

abstraction



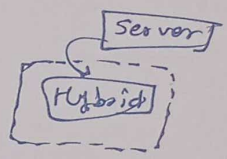
{windows → xamarin.}

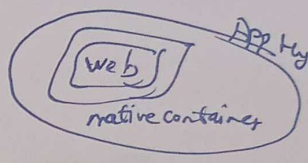
Network H/W also present

① specific to OS
- Android
- iOS

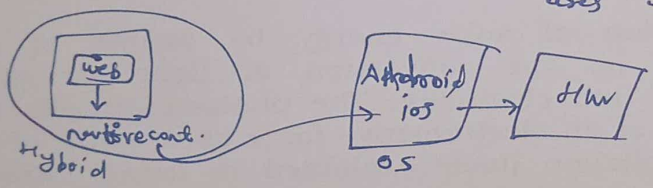
② Hybrid Apps ③ Cross platform

② Hybrid Apps / Web Apps :





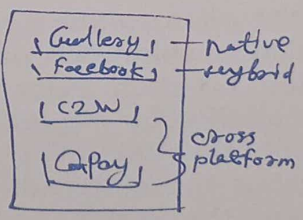
uses web based technologies like html, css, JS



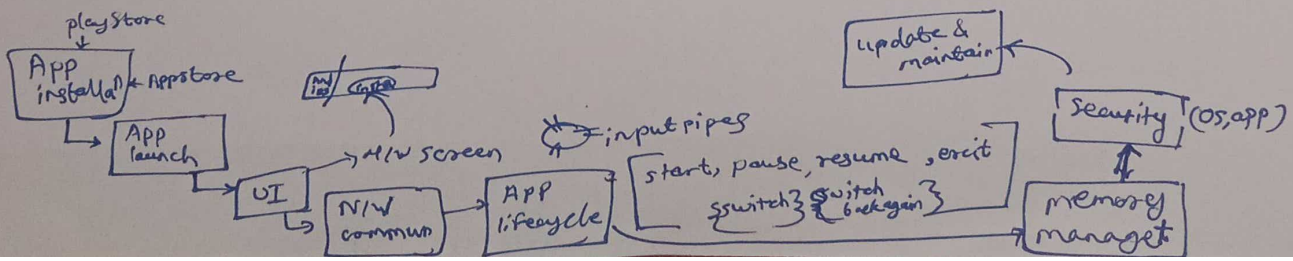
Now comparatively native

③ Cross platform :
- single codebase
- React native, Flutter, Xamarin, Ionic

meta (JS, React) — google (dart) — Microsoft (C#, .net)

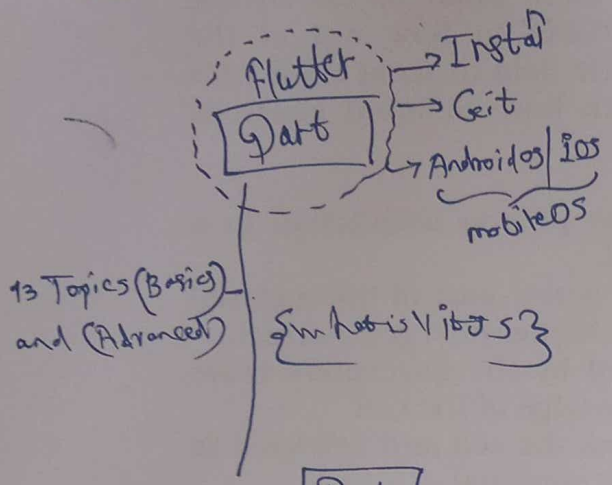


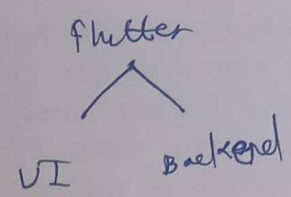
→ rendering engine → make UI glossy

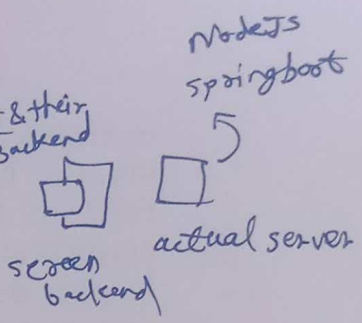


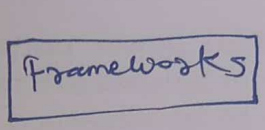
flutter Doctor

[mobile & PC config]



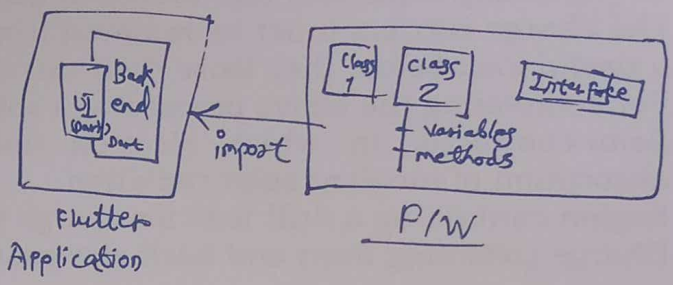






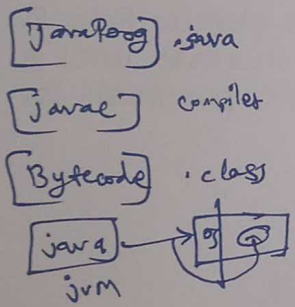
real time playstore local authors

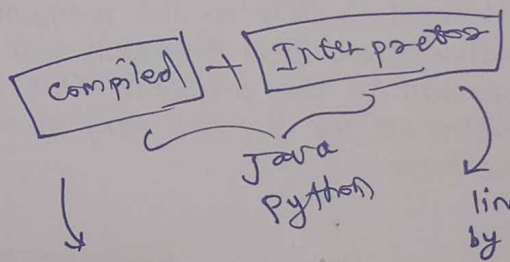
All services Building infrastruct.



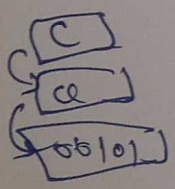
Download
Install
Launch

Introd^n to flutter 8



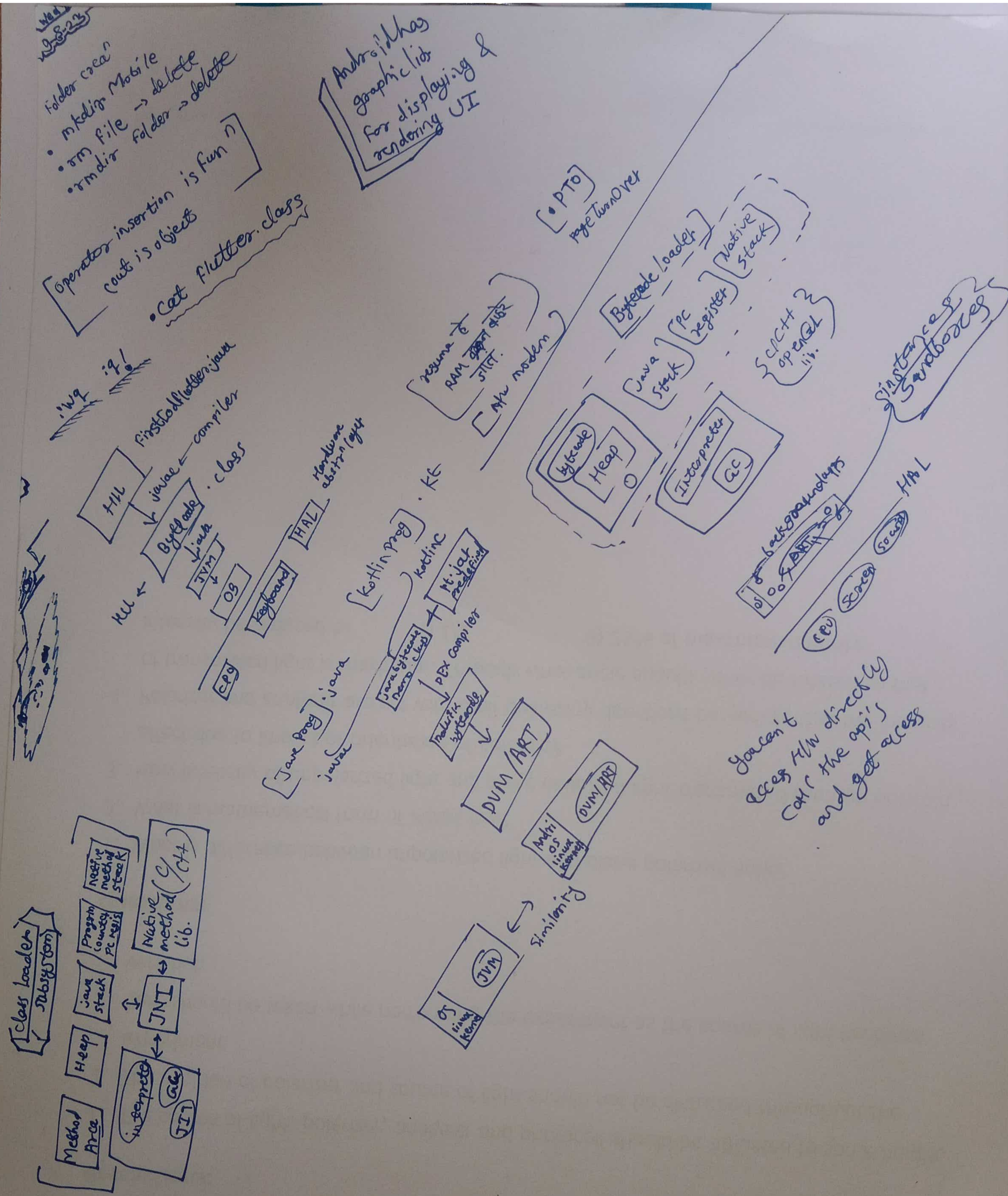


{emulators simulators}



Whose ever OS these App stores is present on mobile which means that closer to compiler

- precompiled & prebuild files on appstores


Wed 9-8-23
folder creation
mkdir Mobile
• rm file → delete
• rmdir folder → delete
Operators insertion is fun
cout is objects
• cat flutter.class
Android has graphic lib for displaying & rendering UI
:wq !
FirstOdFlutter.java
HLL
javac ← compiler
Bytecode
.class
java
JVM
OS
CPU
Keyboard
HAL
Hardware abstraction layer
ML
Kotlin prog
.KE
kotlinc
pt: jar predefined
java bytecode
DEX compiler
dalvik bytecode
DVM/ART
Java prog
.java
javac
Andri OS linux kernel
DVM/ART
similarity
JVM
OS linux kernel
class loader subsystem
Method Area
Heap
java stack
Program counter PC registers
Native method stack
JNI
Native method (C/C++) lib.
interpreter
GC
JIT
RAM
Arm modem
• PTO
page turn over
Bytecode loader
Java stack
PC register
Native stack
C/C++ openGL lib.
bytecode
Heap
Interpreter
GC
instances sandboxes
background apps
HAL
CPU
screen
sound
You can't access H/W directly call the api's and get access

• Javascript: oop based

Dart (Parenthes)

- Basic {DartSDK}
  - ↓ Dart Archi. → Advanced → Flutter archit.
- Advanced → (Flutter + Dart)

Intro. Dart :

Dart → Lars Bak & Kasper

Nov – 2013 : google

→ pure OOP's

• smallTalks

→ C : structural (Pop) Dart : without / with } classes

Same as cpp & python java : without ✗ / with

• Code C.c

```
#include<stdio.h>
  void main(){
     printf("HelloWorld");
  }
```

compiled:
cc codec.c
./a.out

while location finding
• Home direc
• current direct.

• kernel gives access to {JVM–OS–H/W}
kernel

in cpp :
return 0; is returns that value to os
that shows it successfully returns.

• Dart :

CodeDart.dart

```
void main() {
      print("Hello Dart");
}
```

Run:
dart CodeDart.dart
↑
compiler/Interpreter

Dart Code
↓ dart Demo.dart

- exe
- kernel → object
- Desktop → JIT
- Produⁿ (DevOps) → AOT (Ahead of Time)
- web → dartJS (dart to JS)

compilatⁿ ways in dart

its a overloading changed wherever is used.

• That's why it is used multi desktop & mobile OS's.

JS : Dep and maps dependencies

with help node . JS File runs on command line

xyz.exe ↔ specific to windows

Flutter Doctor
occasionally [illegible]

Shodi.com
APP [illegible]
[illegible]

multi files remove

rm codec javaProg pyProg

-> dart run CodeDart.dart

-> dart compile CodeDart.dart

{ dart VM : DVM }

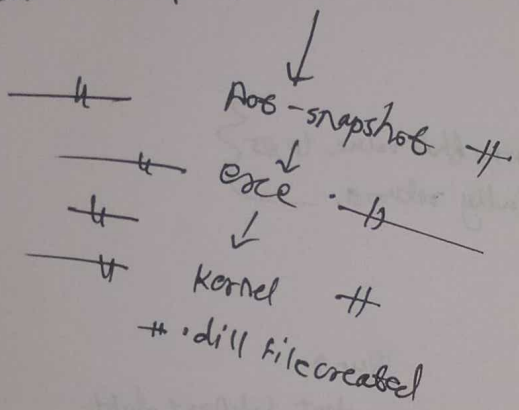
{ dart pad ~~[illegible]~~ } online compiler

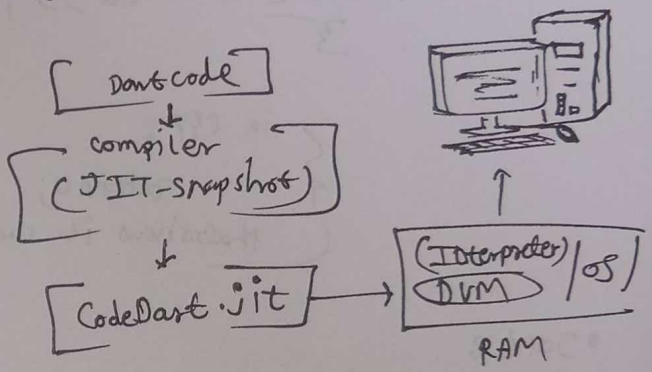
Run: JS : node out.js

by default is JIT

~~dart~~ compile

dart compile jit-snapshot CodeDart.dart





15 Aug 2023
Tuesday

- marshmello onwards
- API level

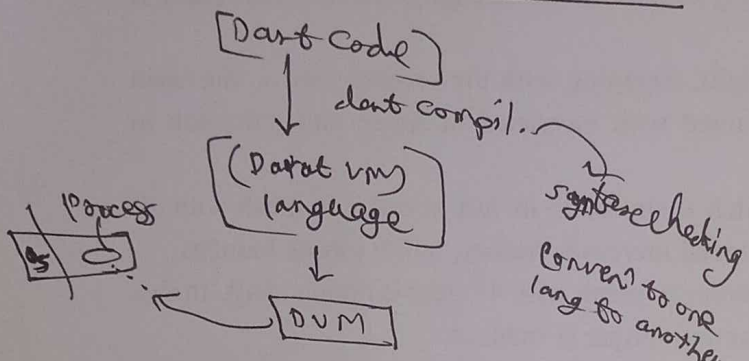
- Thread management is necessary in dart (should be synchronised)

- dart is isolated far from Thread
- Data Types in Dart

```
void main() { print("C2W");   // void without works.
              print("Incubator");
            }
```

Run: dart program.dart.
output: C2W ← 1st line
Incubator ← 2nd line

- It's not need \n for new line.



To Format File:
dart format Prog1.dart

Prog2.dart

```
main() {
  int x = 10;
  print(x);
}
```

output: 10 :

← Entry point fun

index;
// try default
print(x);

output: error: Non-Nullable variable x must be assigned before it used.

- java must be initialized variable here can be unassigned works.

```
num y = 20;
print(y);
```

output: 20

```
y = 70   // Assignment
y = 20.5 // num can be int & double
```

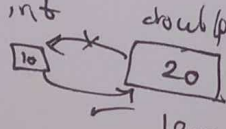
precision point

```
int x = 10; print(x);
num y = 20; print(y);
y = 20.5; print(y);
x = 55.5; print(x); ← error
```

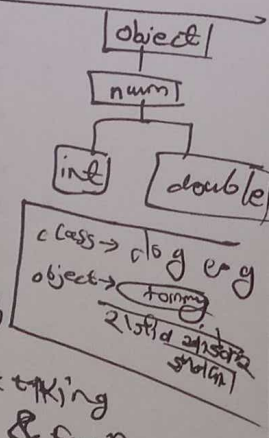
→ A value of type 'double' can't be assigned to a variable of type 'int'.

```
double x = 10.5; print(x);
num y = 20.5;    print(y);
x = 20;          pri--;
y = 30;          --;
```

output: 10.5 ←
20.5 ←
20 ←
30.0



```
double x = 10.5;
print(x.runtimeType);
num y = 20;
print(y.runtimeType);
```



object: it that which needed memory.
↳ dog routine.... walk talking
it's a properties & fun

- own stories, mojo

- int, double, num is classes but treated (looks) like primitive type.
- String is same as java which shows un-Uniformity.

```
String name1 = "c2w";
String name2 = 'Incubator';
print(name1);
--  name2);
```

// UTF-support

- bool flag = true;
  print(flag);

```
main(){
  var X="Crew";
  print(X);
  var y = 20.5;
  print(y);
}
```

var X = "Crew"; ← already assigned first

XrunTime
print —
X = 20.5;
print —
X = true;
print —

output: string can't assigned to double bool

- Twist that why specific is more error free lang.

before (9) → (10)

(lang client)

- fundamentals
  - statement
  - Datatype & variable
  - control-statement
  - I/P, O/P ... due to some prerequisite.
- OOP
  - class & object
  - list + table ... collection

variable-names can be different that's why called identifier.

Operators in Dart

a + b ← operands
↑ operator

operators

- Unary — pre & post
- Binary — a+b, x-y
- Ternary — (x<y) ? x : y (2, 3 operands)

```
main(){
  int x=10;   (-5)
  pri(++x); //11  (-4)
  pri(x++); //11  (-4)
  pri(x);   //12  (-3)
}
```

internally calls Fun^n
prec()
postc()

Q.

```
int x=12;
int ans = ++x + ++x;   (27)  x=14
pri(ans);
ans = --x + --x;       (25)  x=12
pri(ans);
```

- // repeat ++x + x++; (26) x=14;
- // repeat --x + x--; (26) x=12;

dynamic is runtime type modified capable

```
dynamic x=10;
dynamic y=20.5;
print(x.runtimetype);   // int
                        // double
// y.
x=29.7;
y=30;
print(x.runtimetype);   // double
print(y.runtimetype);   // int
```

• Constant variable

int y; ← not allowed :: null safety
↑ i.e initializ^n is needed.

```
Demo obj = new Demo();
obj = null
obj.func();
```
↑ null pointer exception

init^n means value given at time of declaration.

```
main(){
  const int x=10;
  const x=30;

  const int z;   //error
  x=50;          //error
```

- The const variable 'z' must be initialized.
- Can't assign to const variable 'x'.

Arithmatic Operators :

+ - / * % ~/

```
main(){
  int x=10;
  int y=5;
  pri(x+y);
  pri(x-y);
  pri(x/y);
  // (x*y);
  // (x%y);
  // (x~/y);
```

Relational operators

// <, >, <=, >=, ==, !=

false, true, false, true, false, true

logical operators

// !, &&, ||

```
int x=10;
int y=8;
pri(x && y);
pri(x || y);
pri(!x);
pri(!y);
```
} error

That means logical operators required relational operator

exception C

logical operators required boolean in both side

constant : realtime & Rom code but field data.

Blood grp, π, log values.

const ≠ final .... diff things.

- const ↑ compiletime const
- final ↑ runtime const.

```
main(){
  var X="Crew";
  print(X);
  var y=20.5;
  print(y);
}
```

```
var X="Crew"; ← already assigned first
print
X=20.5;
print
X=true;
print
```

XrunTime

Output: string can't assigned to double bool

- Twist that why specific is more error free lang.

22/8/ 12. control statement:

DSA in dart ≈ book

- if statements:

```
if(condition){ ← expression which returns bool values.
  //body
}
```

```
e.g. main(){ int x=7; print x;
     if(x<10){
       print("x is less than 10");
     }
     print("end main");
}
```

o/p: 7
x is less than 10
end main

---

if-else statement:

```
if(condition){
  //body
}else{
  // false if condition
}
```

```
main(){
  int x=5, y=7;
  if(++x == ++y){
    print("same");
  }else{
    print("not same");
  }
  print(x);
  print(y);
}
```

o/p: Not same
6
8

```
main(){ if(++x<=--y)||(x++>++y){
          6     6
    print("same");
  }else{
    print("not same");
  }
  print(x); print(y);
}
```

o/p: same
6
6

{Same condition replace <= , > and || , && }

o/p: Not same
6
6
{6>6 ≈ strictly false}

---

if-else ladder:

```
main(){ int number=7;
  if(number>0){
    print("positive");
  }else if(number<0){
    print("negative");
  }else{
    print("number is zero");
  }
}
```

Realtime:

```
string task="project accomplished";
main(){ if("project accomplished"==task){
    print("promoted");
  }else if(task=="Half done"){
    print("Deadline extended");
  }else{
    print("fired");
  }
}
```

o/p: promoted.

---

looping statement:

```
for(①initialization; ②condition; ④incr/decr){
  //body ③
}
```

```
for(int i=1; i<=4; i++){
  print(i);
}
```

- Murfi bed.

1	i<=4	print(i)	i++
1	1<=4	—	1++
2	2<=4	2	2++
3	3<=4	3	3++
4	4<=4	4	4++
5	5<=4	✗	

x=10
y=8

pri((++x < ++y) && (--x > ++x));
(9) (10) (11)
F && F = F

a=5;
b=6;

(++a < ++b) || (--a > ++b)
6 7 (5) (8)
T || F = T

X = 11 } If one is false in && it return no visit 2nd part
y = 9
a = 5
b = 7 } If one is true in || it return no visit 2nd part

---

leet-11 Bitwise Operator 27/05/23

&, |, ^, <<, >>, ~
↑ bitwise AND, ↑ | OR, ↑ ^ XOR, ↑ left-shift, ↑ right shift, ↑ ~ NOT (negation)

128 = 10000000
130 = 10000010
175 = 10101111
257 = 100000001
85 = 1010101
67 = 1000011
95 = 1011111
72 = 1001000

•Bitwise AND (&)

```
main() { int x=4, y=7;
    print(x&y); //4

  int x=14, y=72;
    print(x&y); //8

  int x=10, y=12
    print(x|y); //14

  print(x^y); //6
}
```

1010
1100

0 | → 1
1 0 → 1
1 1 → 0
0 0 → 0

left shift: x=15, x<<2 = 60
right shift: x>> = 3

43 = 101011 (32 16 8 4 2 1)

x=43
x<<3 = 101011000 (256 128 64 32 16 8 4 2 1)
= 344

y = 72 = 1001000
y>>4 = 100
= 4

Type-checking Operator ***

// as, is, is!

```
int x=4;
main() { print(x is int) ;
  print(y is int)   print(y is num);
  print(z is! int) }
```

Control statement in Dart:
if, if-else, for, while, do-while, switch, break and continue.

---

* pri(x is dynamic) ✓ not class
pri(x is num) ✓
++ (x is var) ✗
++ (x is object) ✓

object
↓
num
int double

int x=4;

③ Loops -

① for loop -

-true -false

#Syntax -

```
for( initialization, condⁿ, inc/dec){
        // body
}
```

(① initialization, ② condⁿ, ④ inc/dec, ③ body)

ⅠⅠ while loop -

#Syntax -

```
① Initialization
while( ② condⁿ){
    ③ // body
    ④ inc/dec
}
```

॥ शुभं ॥

24/08/2023

```
• void main(){
    for(int i=1; i<=20; i++){
        print(21-i);
    }
}
```

```
void main(){
    int n = 20;
    for(int i=n; i>0; i--){
        print(i);
    }
}
```

```
• void main(){
    int n=20;
    for(int i=1; i<=n; i++){
        if(i%4==0){
            print(i);
        }
    }
}
```

✱ Break and Continue -

④ Break -

```
void main(){
    int n=100
    for(int i=1; i<=100; i++){
        if(i==20){
            break;
        }
        print(i);
    }
```

(5) Continue.

```
void main(){
    int i=1;
    while(i!=100){
        if(i%4==0 && i%5==0){
            continue;
        }
        print(i);
    }   i++;
    }
}
```

-
```
void main(){
    int i=1;
    int n=100;
    while(i!=n){
        if(i%2==0){
            continue;
        }
        print(i++);
    }
}
```

-
```
void main(){
    int i=1;
    int n=100;
    while(i!=n){
        if(i==50){
            print(i);
            break;
        }
    }
}
```

lect-14

- nested loops

① nested-for loop -

```
for(initialn ; condn ; inc/dec) {
    for(initn ; condn ; inc/dec) {
        { //body
        }
        {printf(...);}
    }
}
```

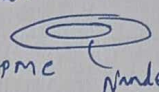
e.g:

```
for(int i=1; i<=3; i++){
    for(int j=1; j<=3; j++){
        printf(j);
    }
}
```

o/p: 1 2 3
1 2 3
1 2 3

but for dart to print everything prints in one line.

Real-time:



dryRun:

i	i<=3	i body	j	j<=3	body	j++
1	1<=3	✓	1	1<=3	✓	2
			2	2<=3	✓	3
			3	3<=3	✓	4
			4	4<=3	X	
2						
3						
4	4<=3	X				

```
main(){ int count = 9;
    for(int i=1; i<=3; i++){
        for(int j=1; j<=3; j++){
            print(count);
            count--;
        }
    }
}
```

```
main(){ int num = 1;
    for(int i=1; i<=3; i++){
        for(int j=1; j<=3; j++){
            print(num);
            num += 2;
        }
    }
}
```

1 3 5 7 9 11 13 15 17

- do-while loop:

{exit controlled loops}

[while loops are entry controlled loops]

→ pub e.g

- Switch statement:

```
int x=10;
switch(x){
    case 1: print(10); break;
    case 2; print(20); break;
    default:
        print("No match");
}
```

// do not required breaks in dart (valid code).
// even if you give break still valid code.

```
main(){
    for(int i=1; i<=3; i++){
        int counter = 1;
        for(int j=1; j<=3; j++){
            print(counter);
            counter++;
        }
    }
}
```

o/p: 1 2 3 4 5 6 7 8 9

dry Run:

	i	i<=3	i body	j	j<=3	body	j++
counter=1	1	1<=3	✓	1	1<=3	1	1++
				2	2<=3	2	2++
				3	3<=3	3	3++
				4	4<=3	X	
counter=3	2	2<=3	✓	1	1<=3	4	1++
				2	2<=3	5	2++
				3	3<=3	6	3++
				4	4<=3	X	
	3	3<=3		1	1<=3	7	1++
				2	2<=3	8	2++
				3	3<=3	9	3++
	4	4<=3	X	4	4<=3	X	

```
var x = "monday";
switch(x){
    case 1: pri...
    case 2: pri...
    default:
        print("No match");
}
```

o/p: No match.

```
var x = 1.0;
switch (x) {
    case 1.0:
        pri—
    case 2.0:
        pri—
    case 3.0:
        pri—
    default:
        pri—
}
```

```
main() { int x = 3;
    do {
        pri—("1:c");
        pri—(2:cpp
        pri— 3:JAVA
        pri— 4:Dart
        switch (x) {
            case 1:
                print 1972
            case 2:
                print 1978
            case 3:
                print 1995
            case 4:
                print 2013
            default:
                pri—
        }
    } while(false);
```

// template
// code

26/08/23 Practical Day:

Q. ① if ((a>b) && (a>c)) {
        print("a is greater");
② } else if (b>c) {
        print("b is greater");
④ } else {
        print("c is greater");
    }

one more else-if case required to check equality case (10,10,10)

③ else-if (a==b && a==c) {
        print("all equal");
    }

Dry Run: In case: 10, 10, 10

① 1st if false
2nd else-if false
3rd is true
then return prints "all equal" and exit if-else ladder.

② (15, 10, 13)
1st if true
prints "a is greater".

③ (10, 5, 15)
all if and else-if false
and that's why prints "c is greater".

- switch can't return anything but in case of dart it is possible
- check version wise changes
- switch also included continue

- For loop also contain label.
- null safetyy feature (← dart)

- duplicate cases allowed in switch
- because it compares objects addr. if it having duplicate values.

---

o/p :→ Hello
dart
Hello flutter

{→ indicate terminal header}

→

Functions:

Types

Userdefined fun.

main() is {}
userdefined

Predefined (C library) fun.

printf(),
write()

util anonymous learn it is compulsary.

until compulsary

optional

returnType FunctionName (parameters){

//body.

}

Types of fun:

1) No parameter - No return value
2) parameters - No ——"——
3) No para - return value
4) parameters - ——"——
   ↳ most used.

Basics of I\O:

C -> #include <stdio.h>
cpp -> #include <iostream>

C in previous iostream.h works now it replace with namespace std;

concept

- (CriminalJustice plot)
- (Group)

- having predefined fun^ but not its declarations.

java -> import java.util.*;

[comment -> //, /**/]

- compiler didn't give exact error's except java which is actual.

{directory means folder}

io is

---

void func() {
  print("In fun");
}

parameter

void main() {
  print("Start");
  Fun();
  print("End");
}

Argument

double parameter → int argument
possible as didn't loss of data.

---

import 'dart:io';

main() {
  print("Hello");
  print("dart");
  stdout.write("Hello");
  stdout.writeln("Flutter");
}

fun1 ← main
fun2 → print
fun3 → stdout.write
fun4 ← stdout.writeln

print terminal in new line.

---

void fun(String name, [double sal = 10.8]) {}

default values for argument.

void main() {
  Fun("Kanha", 20.5);
  Fun("Kanha");
}

named argument

```
void Func({String? name, double? sal}){

}

main() {
  print("main");
  Fun( sal: 20.5 , name: "Ram");
}
```

Error: Too few positional argument : 2
required , 0 given.

- ? allowed & required keyword too

ML-CREVA

16-1eet

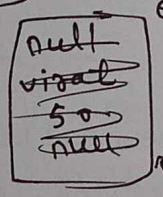
```
void playerInfo({int? jerNO, String? name}) {
    print(jerNO);
    print(name);
}
void main() {
    playerInfo(name: "virat", jerNo: 18);
    playerInfo(name: "Rohit");
}
```

o/p:
18
virat
null
Rohit

```
void main() {int? age = null;
    String? name="virat";
    print(age);
    print(name);
    age = 50;
    null = null;
    print(age);
    print(name);
}
```

int nullable
String nullable

o/p:



error ~~(cannot)~~ possible ∵ we reassign value to it.

write output

//Synchronization
//String interpelation
{data represen under string}

"jerNo is $jerNo"

most used class String

what is json format.

- Errors: Too Few Positional Argument
- Named the parameter.
- try without ? / Required in
- prepare himself for null value as a input.

- IF lang don't allowed us null value that means it follows null safety.
  (2.12 version allow)
- segmentation Fault.
  ↳ supports robust programming

```
void playerInfo({int? ——}, [String team = "india"]) {
}
main() { playersInfo (name: "virat", jerNo: 18);
    ——— (name: Rohit);
    ——— (null, null);
}
```

with named or without named.

---

o/p: Error: Expected ')' before this.
Error: Undefined name 'team'.

- named arg. not required if right seq. given.
- nullable argument compulsary required named arg.

```
playerInfo(String team, {int? jerNo, String? name}) {
  print(++);
}

main() {
  playerInfo("India");
  playerInfo("India", jerNo: 18);
  playerInfo("India", jerNo: 18, name: "virat");
}
```

'required' keyword

```
playerInfo(required String team, ...)
```

Compulsary field.

1-9-23 Lect-17

- return type of fun^n
- lamba fun^n
- anonymous fun^n
- nested fun^n (Inner fun^n)
- recursion fun^n.

e.g.
```
void add(int x, int y) {
}
void main() {
  int sum = add(5, 10);
  print(sum);
}
```
① ←

o/p : error - This expre^n has type 'void' and can't be used.

If
```
int add(int x, int y) {
}
```
error: non-null value not return null value.

```
int add(int x, int y) {
  return x+y;
}
void main() {
  print(add(x, y));
}
```
1st way.

Data discarded. not error

* code works correctly.

c-printf returns value but we don't add container

2nd way
```
int retVal = add(10, 20);
print(retVal);
```

Lambda fun^n or arrow fun^n:

- same name of fun^n, parameter, return type generates ambiguity.
- other prog. lang. gives 'Note' dart gives 'Context'.

→ same fun^n and called it as
```
print(add(10, 20));
```
generates 3 error as previous declarat^n and 1 for print(add(10, 20)); line.

```
int div(int x, int y) {
  return x/y;
}
void main() {
  print(div(10, 10));
}
```
→ error: 'int' type of return type can't be converted into 'double' type.

- data loss can't allowed in dart.

so change it as num or dynamic or double or var.

✡

- Arrow function limitcal cases nadheek work hoto!

```
int add(int a, int b) => a+b;
int div(int a, int b) => a / b;   // double value
void main(){
    ---------
    ---------
}
```

← In arrow fun^n depends output on prototype parameters and return type

- you can write point also infront of =>.

← didn't give error.

← return Type void.

① single line

✡ Recursion function:
fun^n in itself call the fun^n recursively

```
void fun(){
    print("In fun");
    fun();
}
void main(){
    fun();
}
```

o/p: error: ~~call recursively~~ / ~~or infinite loop~~. (mine)

actual -> Infinite loop —then→ unhandled except^n: stack overflow.

[delayentoypointEnvocat^].

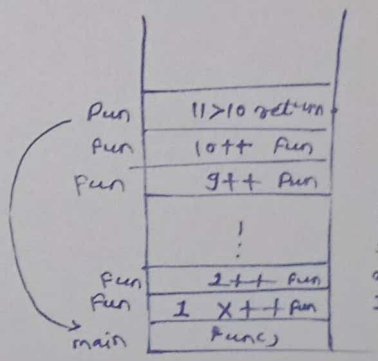
- In Stack 'LIFo' is recommended way rather FILO.
- Iterative vs Recursive

```
void fun(int x){
    if(x>10){
        return;
    }
    print(x++);
    fun(x);
}
void main(){
    fun(1);
}
```

```
void fun(int x){
    if(x>10){
        return;
    }
    print(x++);
    fun(x);
}
void main(){ int x=1;
    fun(x);
}
```

- for no-parameter fun() try x as a glob variable.

```
int x=1;
void fun(){
    if(x>10){
        return;
    }
    print(x++);
    fun();
}
void main(){
    fun();
}
```



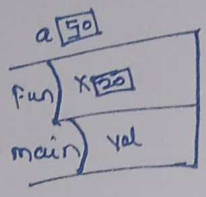
o/p: 1 to 10

RiverOS Usage [NEED] | AOVP The smart seat

Leet - 18

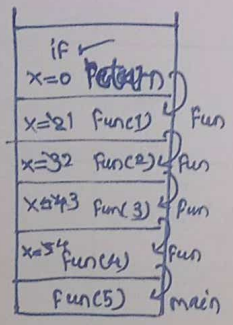
```
int a=50;   //global -> sum&mean
int fun(int x) {
    print("In fun");
    a = a+x;
    return x;
}
void main() {
    print("start main");
    int val = fun(20);
    print(val);
    print(a);
    print("End main");
}
```

o/p : start main
In fun
~~20~~
~~30~~
End main



Recursion :

```
Void fun(int x) {
    if(x<=0) {
        return;
    }
    print(x--);
    Fun(x);
}
Void main() {
    print("main");
    Fun(5);
    print("End");
}
```

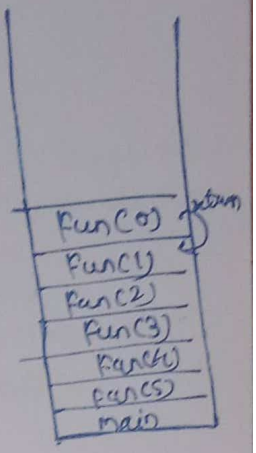


- 1st time if we run code as dart prog1.dart it takes time to call virtual machine.

```
int printSum(int x) {          //for loop
    int sum=0;
    for(int i=1; i<=x; i++) {
        sum += i;
    }
    return sum;
}
void main() {
    print(printSum(5));
}
```

- with Recursion :

```
int sum=0;
void printSum(int x) {
    if(x==0) {
        return;
    }
    sum += x;
    x--;
    printSum(x);
}
void main() {
    printSum(5);
    print(sum);
}
```



- with loop : factorial

```
int factorial=1;
void printfactorial(int num) {
    for(int i=1; i<=num; i++) {
        factorial *= i;
    }
}
void main() {
    printfactorial(5);
    print(factorial);
}
```

- with recursion :

```
int factorial=1;
void printfactorial(int num) {
    if(num==0)
        return;
    factorial *= num;
    num--;
    printfactorial(num);
}
void main() {
    printfactorial(5);
}
```

Total 7 Stack frames. then return out fm.

Lect-19

Lambda Fun^n & anonymous Fun^n:

- In Lambda Fun^n its object get created and we used to call that object.

```
var add = (int a, int b) {
        return a+b;
};
void main() {
        print(add(10,20));
}
```

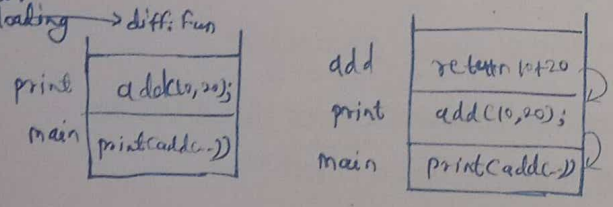
object → add; class → { }; Try without ;

- stack from start with main to main its called normal terminatn. else it called abnormal terminatn.

- It's not like normal Fun^n as its stack-frame pushes, in case of lambda add is object and from {} its a class which its stack frame is not going to push.

∴ object(10,20) is there but object can't be parenthesis
↳ internally it will call differently that means ∧ operator overloading is performed.

i.e operator overloading → diff: Fun



- write upper instance in main.

```
void main() {
        var add = (int a, int b) {
                return a+b;
        };
```

add का object ka define करायेगा हो value देगा तभी.

Try dynamic? & var?

```
void main() {
        var add = () {
                print("Hello Object");
        };
        print(add());
}
```

o/p: Hello object
null

2-Things return by lambda Fun^n.

```
print(x.runtimeType);      //int
print(add.runtimeType);    //() => void
```

o/p: int
() => null

(int, int) => int

Works!

↳ for this o/p:-

```
var add = (int a, int b) {
        print("Hello");
        return a+b;
};
```

```
void main() {
        (int a, int b) {
                print("Hello");
                print(a+b);          //return not possible
        }(10,20);
}
```

o/p: Hello
30

- same as anonymous inner class in Java.

↳ Try print(10,20) in this position.

```
void main() {
        var printData = () => print("In Fun^n");
}
```

o/p: In Fun^n.

Input / Output in Dart :

```
import 'dart:core';
```
default package not need to be imported just like lang package in Java.

stdin is object
class is S ← checkout

```
void main() {
  print(stdin.runtimeType);
  int age = stdin.readLineSync();
  print("Age = $age");
}
```
called string interpolation

//class Stdin

its return value is - String?

async → ~~Future वाला फंक्शन~~

```
:. String? name = stdin.readLineSync();
```

```
int? age = int.parse(stdin.readLineSync()!);
```
given input as string.

gives exception
int._handleFormatError.
& Stacktrace.

Task - 20

```
void main() {
  int x = 10, y = 20;
  var FunObj = (int a) {
    print(a);
    print(x);
    print(y);
  };
  FunObj(30);
}
```
← Not work as a class instance

o/p: 30
10
20

```
import 'dart:io';
void main() {
  int? empId;
  String? empName;
  double? sal;
  print("Enter employee ID:");
  empId = int.parse(stdin.readLineSync()!);
  print("Enter employee Name:");
  empName = stdin.readLineSync();
  print("Enter employee Salary:");
  sal = double.parse(stdin.readLineSync()!);
}
```

```
stdout.write("Id: $empId, Name: $empName,
  Sal: $sal \n");
```
shown differently

works!, but recommended way is writeln()

- persistant founder
- brochure.

lect-20

```
class Player {
    int jerNo = 7;
    String pName = "MSD";
    void playerInfo() {
        print(jerNo);
        print(pName);
    }
}
void main() {
    Player obj = new Player();
    obj.playerInfo();
}
```

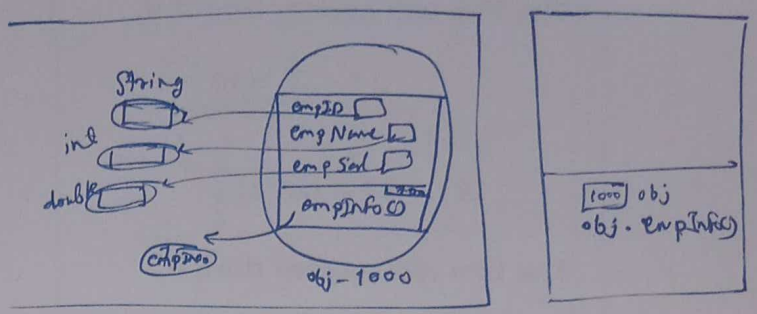
O/P: 7 MSD

```
obj.jerNo = 45;
obj.pName = "Rohit sharma";
obj.playerInfo();
```

O/P: 45 Rohit Sharma

```
import 'dart:io';
class Employee {
    int? empId = int.parse(stdin.readLineSync()!);
    String? empName = stdin.readLineSync();
    double? sal = double.parse(stdin.readLineSync()!);
    void EmployeeInfo() {
        stdout.writeln("$empId, $empName, $sal");
    }
}
void main() {
    Employee obj = new Employee();
    obj.EmployeeInfo();
}
```

Runtime Memory Area



Real Time

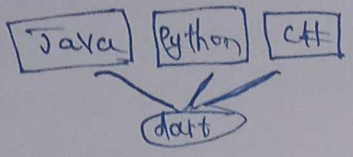
```
import 'dart:io';
class Country {
    String? cName = "Unknown";
    int? noOfStates = 0;
    double? gdpRate = 0;
    void cInfo() {
        stdout.writeln("Country Name: $cName
            \nNo of states: $noofstates.
            \nGdp Rate: $gdpRate");
    }
}
void main() {
    Country c = new Country();
    print("Enter Country Name:");
    c.cName = stdin.readLineSync();
    print("Enter No. of states:");
    c.noOfStates = int.parse(stdin.readLineSync()!);
    print("Enter Country growth rate:");
    c.gdpRate = double.parse(stdin.readLineSync()!);

    c.cInfo();
}
```

lect-21

[operator()]

↳ callable function Type

internally of lambda function.

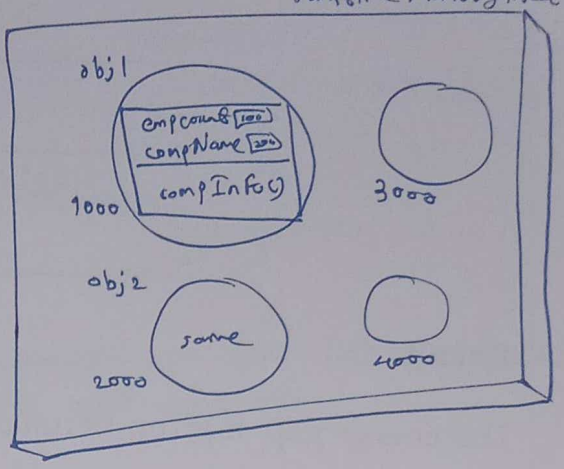



```
class Demo {
        int x = 10;
        String str = "class Object";

        void info() {
                print("$x $str");
        }
}
void main() {
        info();
        print(x);
}
```

o/p: Error - Method not found: 'info'.
Error - Undefined name 'x'.

- can be possible with creating objects.

RMA
runtime memory Area.




```
class Employee {
        int empId = 10;
        String empName = "Ashish";
        double sal = 1.0;

        Void empInfo() {
                print(——
        }
}
void main() {
        Employee obj1 = new Employee();
        obj1.empInfo();
        Employee obj2 = Employee();
        obj2.empInfo();
}
```

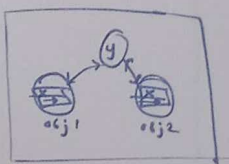
class Variables & instance Variables

```
Class Company {
        int emCount = 50;
        string compName = "google";
        string loc = "pune";
        void info() {
                print(——
        }
}
void main() {
        company obj1 = new company();
        obj1.info();
        company obj2 = company();
        obj2.info();
        new company(); .info();
        company(); .info();
}
```

↳ 4 Types of object creation.

class Variable or static Variable

```
class Demo {
        int x = 10;
        static int y = 20;
        void printData() {
                print(——
        }
}
void main() {
        //same as Employee section
}
```




```
obj1.y = 50;
obj1.printData();
obj2.——;
```

10
50
10
50

Error: ~~Setter~~ 'y' isn't defined for the class 'Demo'.

∴ Demo.y = 50;

Lect-22

- dart is having isolated box like struct.
  ↳ separate ecosystem behaviour

class ला memory असते.
{ Instance is object variable.
  static is class variable. }

```
import 'dart:io';
class Demo {
    int? x = int.parse(stdin.readLineSync()!);
    static int? y = ————————//
    void printData() {
        print( ————
    }
}
void main() {
    Demo obj = new Demo();
    obj.printData();
}
```

memory?

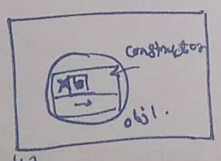
o/p: 10
10 ←
20
20 ←

sync → one by one execution.

If obj creation line is commented we don't get error.

• Constructor

```
class Demo {
    int x = 10;
    Demo() {
        print("constructor");
    }
}
void main() {
    Demo obj = new Demo();
}
```

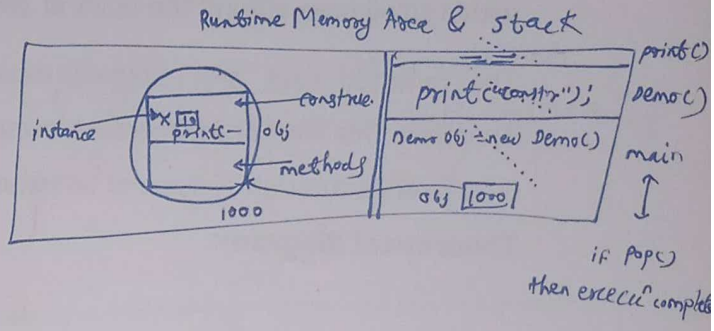


- class & constr same name.
- Doesn't have return value.
- No need to call auto call when obje. created.
- No arguments, default by dart, parameterized. compiler.
- No need to call explicitely.

```
class Demo {
    void Demo() {
        print("constructor");
    }
}
void main() {
    Demo obj = new Demo();
}
```

Java मध्ये चालतो कारण तिथे method च असते इथे सगळे objects आहेत.

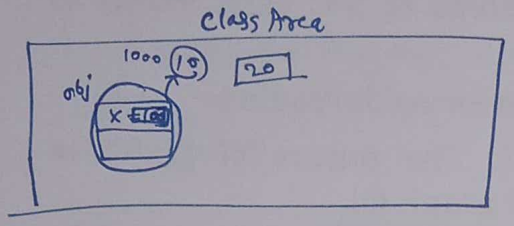
Error: Constructors can't have return type.



• this reference

Demo(obj) ⟶ Demo(Demo this) {
}

```
class Demo {
        int x = 10;
        static int y = 20;
}
void main() {
        Demo obj = new Demo();
        print(obj.x);   //10
        print(obj.y);   //error
}
```



[No scope to reach 20 like in java there is ptr. to spe. str.]

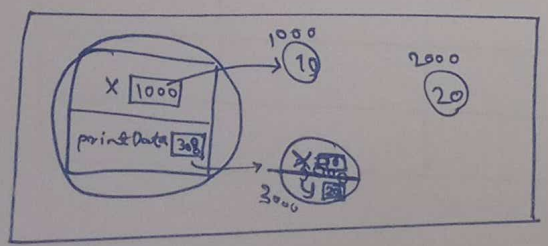
∴ Demo.y ⟶ can be valid.

[non-static called instances.]

[instance methods put on under obj but it will run on stack (RAM)]

∈ it need to call / invoke to run on stack.

```
class Demo{
        int x=10;
        static int y=20;
        void printData(){
                print(x);
                print(y);
        }
}
void main() {
        Demo obj = new Demo();
        printData();   //error: method not defined.
}
```




```
int x=10, y=10;
```

it self is class → x [10] y → same address.

[akasa airlines, read about samsung]

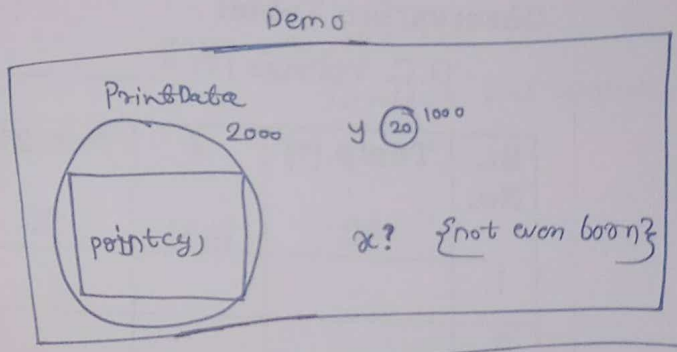
colombia → billionier person.

[static ला obj मध्ये जाता येत.]

[everything isolate in dart]

IF we do →
```
static printData(){
        print(x);
        print(y);
}
```

- compiletime error: 'x' is undefined.
  [Due to is non-static (instance).]



- Constructor - [Born story]

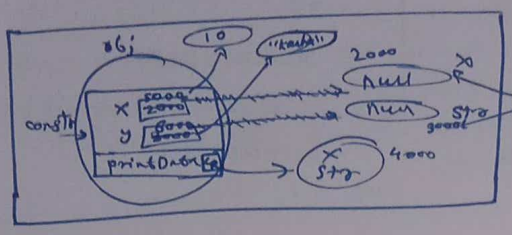
{ in current version null is class whose objects is not parents. [only] }

int? ← it says 'I support null safety'.

```
class Demo {
        int? x;
        String? str;
        void printData(){
                print(x);
                print(str);
        }
}
void main() {
        Demo obj;
        obj.printData();
}
```

o/p: null null

lect - 23




```
obj.x = 10;
obj.y = "kanha";
```

↖ Rather than doing this it's better to be passed by arg. in constructor

↙

- no having return Type
- same as class name
- No need to call explicitly.

```
class Demo {
    int? x;  String? str;
    Demo(int value, String name) {
        print("In para constr");
    }
    void printData() {
        print("$x & $str");
    }
}
void main() {
    Demo obj = new Demo(10, "kanha");
    obj.printData();
}
```

```
Demo(int x, String str) {
    x = x;
    str = str;
}
```

↖ not gives error

o/p is null null.

∴ this.x = x;
this.str = str; ↖ It works!

```
Demo obj = new Demo();   //Demo(obj)
print(obj);
print(obj.hashcode);
print(identityHashCode(obj));
```

both same

↖ try in Java!

//this is hidden parameter comes implicitly.

```
Demo() {
    print(this.hashcode);
}
```

```
int x = 10;
int y = 10;
print(x.hashcode);
```

// cg.

```
int y = new int(10);
```

↖ couldn't find constructor.

26-9-23

lect-24

- const. comes into the picture to initialize instance variable.
- Even constructor is also run on stack.
- If u write parameterized const. default const. do not invokes.

? factory constructor → in dart

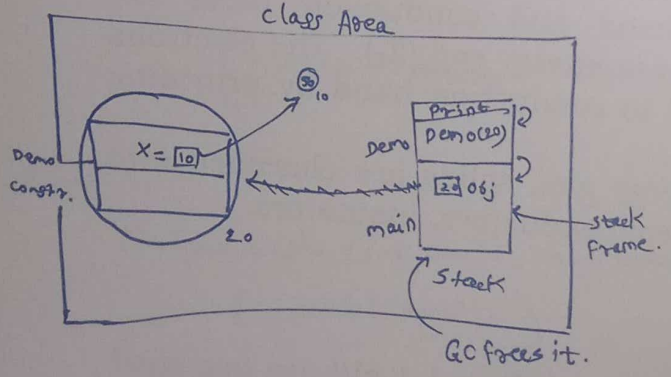
- const. is special method - ()

[ID - identification no.]

[1st time compilat° - PVM, JVM, DVM takes time to load]

- this is variable so DataType is needed.

```
class Demo{
        int x=50;
    Demo(){                    // Demo(Demo this)
        printCx);
    }
}
void main(){
        Demo obj = new Demo();
        //Object create
        // add copy=obj      //100
        //Demo(obj);         // Demo(100)
```



- Standard Way in Dart:

```
Class company{ int? empCount; String? compName;
    Company( this.empCount, this.compName);
}
```

[WAYS of CONSTRUCTOR]

- Default Argument (Parameter) Value Constructor {It's optional}

```
Company(this.empCount, [this.compName="Biencaps"]);

void main(){
    Company obj = new Company(25);
}
```

- Default are compulsory.

```
Company(this.empCount, {this.compName="Biencaps"});

Company obj = new Company(25, "veritas");
                                  {
                                  -> got error
```

//named Parameter.

```
Company({this.empCount, this.compName});

company obj= new Company(compName:"veritas",
                         empCount:25);
```

[Types of constructor]

```
class Employee{
        int? empId; String? empName;
    Employee(){
    }
}
```

{If compiler given constr. is default, If we write Employee(){} is no-argument}

[but in dart is called default & no-args also]

```
//Parameterized constructor.
    Employee(this.empId, this.empName);
```

→ dart prog.java (try it!)

• Named Constructor :

In Java:

```
Demo(){
}
Demo(int empId, String empName){
}
```

(works but not in dart)

{∵ everything is object & same object is not allowed due to ambiguity.}

• error: employee is already declared in scope.

Can't use 'Employee' because it is declared more than once.

• Same name of object not allowed.

Employee.constr = new Employee.constr(25, "Ram");

↳ check error

[^^^^^^ → speedly (Focus Here!)]

For:

```
Employee.constr(int empId, String empName)
{ print("para"); }
```

• constant constructor :

```
{(const Player(this.jerNo, this.pName);
}
void main(){
}
```

error: constructor is marked 'const' so all fields must be final.

---

const and final are quite similar not about same scenarios for private. ← left 25

[• A const constructor can't have body. Remove 'const' keyword or remove body.]

(error

const vs final

• const can't assign to a variable. x = x+1;

```
void main(){
  const int? x;
  print(x);
}
```

error: const value 'x' must be initialized.

{Try: const int? x = null;

• null is class whose parent is not obj.}

• Data must be given compile-time. (const.

• Final variable also can't assign
• final variable must be assign before it can be used.
• Data must be given compile or run-time.

[int.parse(—!);
(can't be null i.e[!]]

• write both codes.

```
import 'dart:io';

class Demo {
        final int? x;
        final string? name;

  const Demo (this.x, this.name);

  }
Void main() {
        int? a= int.parse(stdin.readline()!);
        String? name = stdin.readline();

      Demo obj = new Demo(a, name);
  }
```

-> with the help of const constructor we can create const obj to avoid multiple obj creation.

- Demo obj1= new Demo (10,"Ashish");
  Demo obj2 = new Demo (10, "Ashish");
  ↳ both obj's hashcode() are different.

can't both

Demo obj1 = const new Demo(10,"ram");
— " — obj2 ————— " —————

both obj's hashCode() are same.

- It required when multiple object creation done & same data is passed पण मोठ्या फ़ायदा बनतो.

-> Setter & getter method in dart <-

If fields are final.
and obj.x=10; & obj.y="Hello";

got error:

- dart is not having public, private like keywords.

- Private access specifier:

  int? _jerNo;
  string? _pName;

If we can't write anything it behaves like public access specifier.

[Technical word: syntactical sugar
        big concept bind in a smaller one.]

In Dart: Private access in not between class but in between files.

- Zoriant company.

  [cp prog3.dart prog5.dart.]
        ↳ cmd.

import 'prog3.dart';
  i.e files are libraries.

[cat prog3.dart prog5.dart]
        ↳ cmd
  - make files combines to display purpose.

```
class Player {
        int? _jerNo;  string? _pName;

        player(this._jerNo, this._pName);
        void playerInfo() {
                print(_jerNo);
                ————
        }
  }
```

```
→ import 'prog3.dart';
  void main() {
        player obj = new play——
        obj.playerInfo();
        obj._jerNo = 18;
```

lect - 26

- explicit is user data, implicit is gerven by compiler.
- in import 'prog.dart';
  (indicates same directory we can give full path also.

•Inheritance Real Time

ICC → BCCI → IPL

File 1 [getter methods for private]

```
int? getX() {
    return _x;
}
double? getSal() {
    return _sal;
}
```

File 2

```
print(Obj.getX());
print(Obj.getSal());
```

[→ To bind files import line is IMP.]

way 2 to write getter (get keyword without ())

File 1

```
int? get getX {
    return _x;
}
double? get getSal {
    return _getSal;
}
```

File 2

```
print(Obj.getX);
print(Obj.getSal);
```

[•SEBI full form & work?]

→ prototype ←

Closure: () => void From function 'disp':.

File 1:

```
void disp() {
}
```

File 2:

```
print(Obj.disp);
```

---

way 3:

File 1:

```
get getX => -x;
```

[arrow fun" without return value

```
get getSal {
    return _sal;
}
```

Error: A getter can't have formal parameter (....)

```
if we do - get getSal() {
    return _sal;
}
```

→ get keyword doesn't required parenthesis () it will implicitly handle things.

```
void fun() {}
print(fun.runtimeType);
```

O/P: () => void

- fun() is also object but it's stackframe push due to its callable fun()

① • Setter :

File1 :

```
void setX(int x){
    _x = x;
}
void setName(String name){
    str = name;
}
```

File2 :

```
obj.setX(15);
obj.setName("Ram");
obj.disp();
```

---

② File1 :

```
set setX(int x){
    _x = x;
}
set setName(String name){
    str = name;
}
```

File 2 :

```
obj.setX = 15;
obj.setName = "Ram";
obj.disp();
```

---

③ File1 :

```
set setX(int x) => _x = x;
set setName(String name) => str = name;
```

File 2 :

```
obj.setX = 15;
obj.setName = "Ram";
obj.disp();
```

• Swap between their calls and definition

INHERITANCE

```
class Parent {
    int x = 10;
    String s = "surname";
    void disp(){
        print("parent method");
    }
}
class child extends Parent {
}
void main(){
    child obj = new child();
    print(obj.x);
    obj.disp();
}
```

```
class BCCI {
    int noplayers = 20;
    void matchTypes(){
        print("Test/T-20");
    }
}
class IPL extends BCCI {
    void matchTypes(){
        print("T-20");
    }
}
void main(){
    child obj = new child();
}
```

Leat-27

```
class Parent {
        int x=10; String s1 = "Par";
/*static*/ void parentMethod() {
                print(x);
                print(s1);
        }
}
class child extends Parent {
        int y=20; String s2 = "child";
        void childMethod() {
                print(y);
                print(s2);
        }
}
void main() {
        Parent obj = new Parent();
        print(obj.y);
        print(obj.s2);
        obj.childMethod();
}
```

3 error.

Error: getter '_' isn't defined for the class 'Parent'.

method '___' isn't defined

- gives getter error for instance variables rather isn't defined.

★ • Protocol is do not directly access class content use getter-setter.

- static असेल non-static access hot nahi Karan static सर्वात आधी येतो आणि instances object create झाल्यावरच येतो.

- Error: Member not Found : 'Parent.parentMetho

```
main() { Parent obj1 = new Parent();
        3-ele  obj1.x.s1. parentMethod()
                        accessible.
        child obj2 = new child();
        6-ele obj2.x.s1.y,s2, parent childMethods
                accessible. }
```

```
Parent {
        get getX => x;
        get getS1 => S1;
}
child {
        get getY => y;
        get getS2 => S2;
}
main() {
        child obj = new child();
        print(obj.getX);
}
```

```
class Parent {
        int x=10;  String str1 = "name";
        void parentMethod() {
                print(x); print(str1);
        }
}
class child extends Parent {
        int x=20; string str1 = "data";
        void childmethod() {
                print(x); print(str1);
        }
}
void main() {
        child obj = new child();
        print { obj.x
                obj.str1
                obj.parentMethod
                obj.childMethod }
```

20
data
20 data.
20 data.

- constructor सोडून सगळं child कडे inherit होत
- Dart having seporate concept For constructor inheritance.
- Protected class या Parent class object class aahe.

```
Parent {
        Parent {
            print C "parent const ");
        }
}
child extends Parent {
        child{
          printC "child const");
        }
void main() {
        Parent obj1 =new Parent();
        child obj2 = new child();

        printC obj2.x);
        printC obj2.str2);
        print Cobj2.parentMethod());
}
```

o/p: Par Const
Par const
child const
20
data
20
data

• constructor fakt accessible agto.

---

lect-28.

```
class Parent {
        int x=10;
        void printData() {
            sop(x);
        }
}
class child extends Parent {
        int x=20;
        void dispData() {
            sop(x);
        }
}
class client {
        main() {
            child obj = new child();
            obj.dispData();
            obj.printData();
        }
}
```

o/p:

Draw JVM Archi dig.

```
class Parent {
        int x=10;
        void printData() {
            print(x);
            print(this.hashcode);
        }
}
class child extends Parent {
        int x=20;
        void dispData() {
            print(x);
            print(this.hashcode);
        }
}
void main() {
        child obj = new child();
        obj. ~~4~~ ;
             ~~11~~ ;
```

o/p: In Parent
~~add~~
child const
~~add~~
20
20

```
class Parent {                          // base, superclass, Parent
        Parent {
            print("Par const.");
        }       call() { print("call meth"); }
}
class child extends Parent {            // child, derived, sub class
        child { // super();              // superclass has no method named 'call'.
            print("child const");
        }
}
void main() {
        child obj = new child();
        // obj();
            { errors: The method 'call' isn't defined for class 'child'.
}
```

- call method inherits from parent and gets called implicitely.
- obj(); we calling object to make this object callable we create call method.